Национальный исследовательский университет "МЭИ"

Релейной защиты и автоматизации энергосистем

Лабораторная работа № 1

«МЕТОД К-БЛИЖАЙШИХ СОСЕДЕЙ. ЛОГИСТИЧЕСКАЯ РЕРГЕССИЯ»

Выполнил:	Энтентеев А.Р.
Группа:	Э-13м-19
Проверил:	Нухулов С.М.

Москва, 2020

Лабораторная работа №1.

МЕТОД К-БЛИЖАЙШИХ СОСЕДЕЙ. ЛОГИСТИЧЕСКАЯ РЕРГЕССИЯ

Выполнил Энтентеев Айдар гр. Э-13м-19

In [1]:
```
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
import numpy as np
```

Загрузка данных для обучения.

In [5]:
```
data = pd.read_csv("breast_cancer.csv")
data.head()
```

Out[5]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	poin
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	

5 rows × 33 columns

Подготовка данных для обучения. Удаление лишних не информативных столбцов

In [75]:
```
data = data.drop(columns = ["id","Unnamed: 32"])
data.head()
```

Out[75]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	s
0	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	
1	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	
2	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	
3	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	
4	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	

5 rows × 31 columns

Выделение из общего массива данных массива ответов и выборки для обучения

In [76]:
```
X = data.drop(["diagnosis"],axis =1)
y = data["diagnosis"]
y = pd.get_dummies(y)["B"]
X.head()
```

Out[76]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_m
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1

2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1

5 rows × 30 columns

Разделение выборки на обучающую и тестовую, на которой будет проверяться качество алгоритма

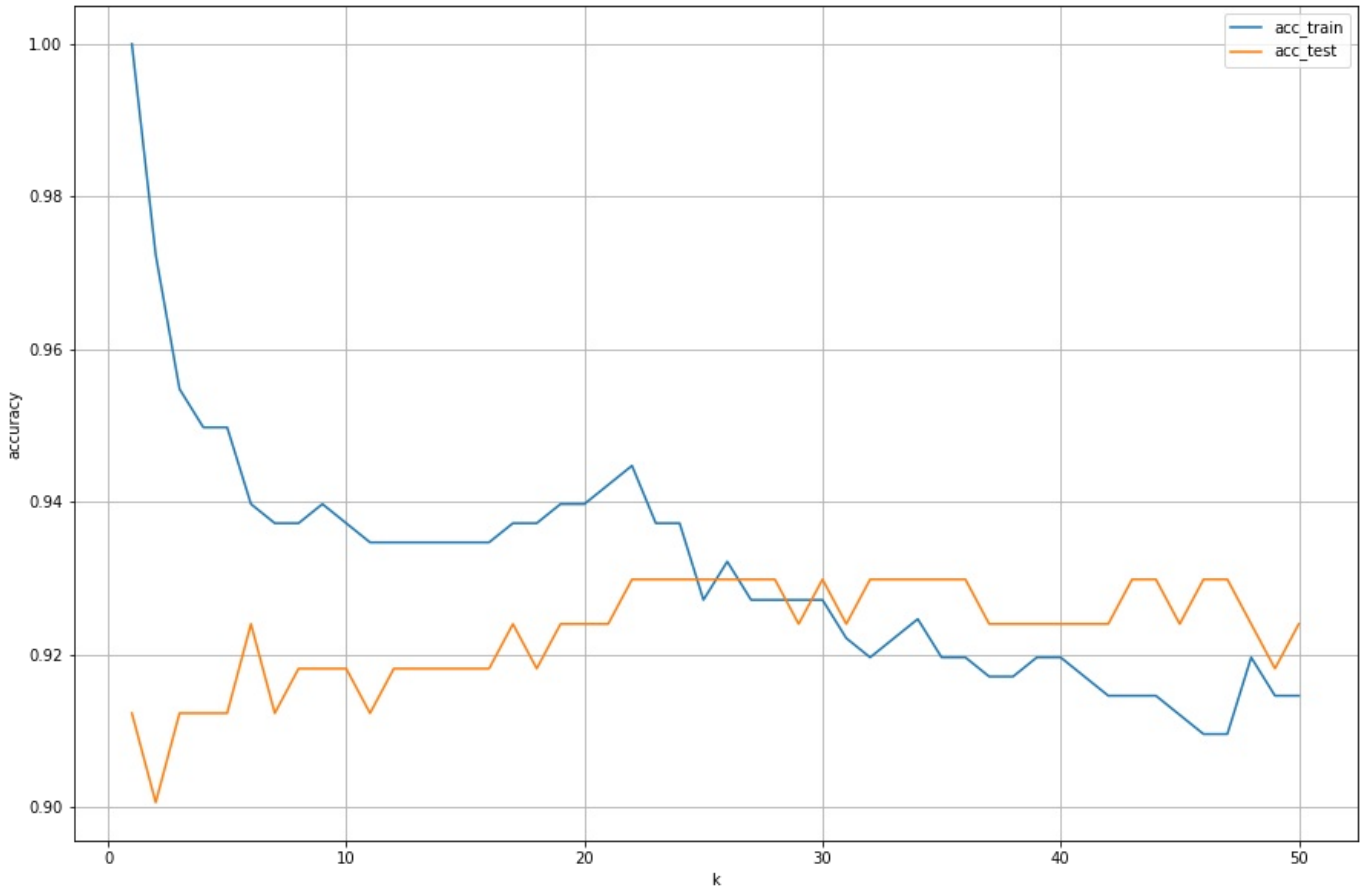
In [77]:
```
X_train, X_test, y_train, y_test  = train_test_split(X,y,test_size =0.3, shuffle = True)
```

Обучение модели и определение качества алгоритма (accuracy) на тестовой и обучающей выборке

In [78]:
```
accuracy_train = []
accuracy_test = []
for k in range(1,51):
    model = KNeighborsClassifier(n_neighbors=k)
    model.fit(X_train,y_train)
    accuracy_train.append(model.score(X_train,y_train))
    accuracy_test.append(model.score(X_test,y_test))
```

In [79]:
```
neighbours = range(1,51)
plt.figure(figsize = [15,10])
plt.plot(neighbours,accuracy_train)
plt.plot(neighbours,accuracy_test)
plt.grid("on")
plt.xlabel('k')
plt.ylabel('accuracy')
plt.legend(["acc_train","acc_test"])
```

Out[79]: <matplotlib.legend.Legend at 0x7fb051157890>



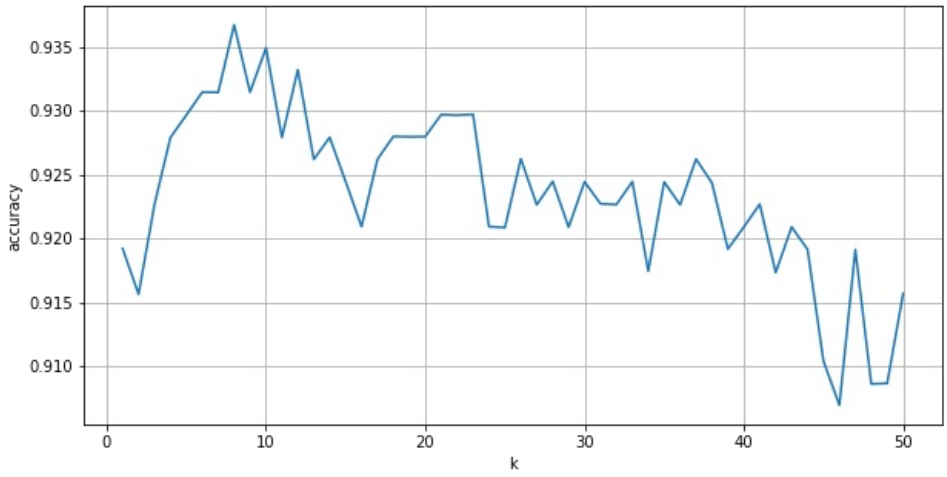
Вывод из графика

Из графика видно, что при проверке точности на обучающей выборке при k = 1, алгоритм дает 100% точность за счет того, что при проверке этот объект не удаляется из общей выборки и он является соседом самого себя. При увеличении же количества соседей происходит уменьшение качества и значения на тестовой и обучающей выборке приходят примерно к одному истинному значению точности.

Обучение модели и проверка качества методом кросс-валидации.

In [80]:
```
kf = KFold(n_splits=5,shuffle=True)
score = []
accuracy = []
for k in range(1,51):
    model = KNeighborsClassifier(n_neighbors=k)
    score = cross_val_score(model, X,y, cv=kf, scoring = "accuracy")
    accuracy.append(score.mean())
```

In [81]:
```
neighbours = range(1,51)
plt.figure(figsize = [10,5])
plt.plot(neighbours,accuracy)
plt.xlabel('k')
plt.ylabel('accuracy')
plt.grid("on")
```



Вывод из графика

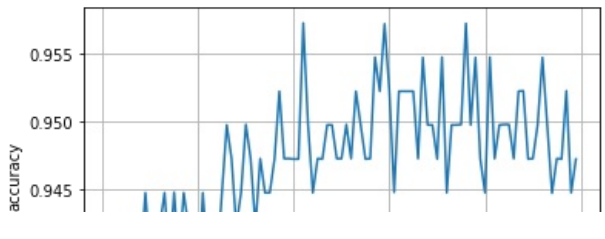
В данном графике отображена точность алгоритма проверенная методом кросс-валидации. Из графика можно заключить, что оптимальным является k ~ 10

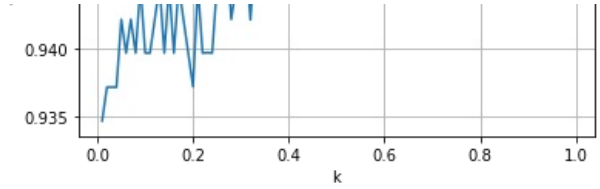
Подготовка данных для обучения методом логистической регрессии.

In [ ]:
```
c = {'C': np.arange(0.01,1,0.01)}
model = LogisticRegression(random_state = 241)
gs = GridSearchCV(model, c, scoring='accuracy', cv=kf)
gs.fit(X_train, y_train)
```

In [55]:
```
plt.plot(c_range, gs.cv_results_['mean_test_score'])
plt.xlabel('c')
plt.ylabel('accuracy')
plt.grid('on')
print("Лучшее значение C: ",gs.best_params_)
print("Лучшее занчение точности: ",gs.best_score_)
```

```
Лучшее значение C:  {'C': 0.42000000000000004}
Лучшее занчение точности:  0.9573101265822783
```





Вывод из графика

В данном графике отображена точность алгоритма проверенная методом кросс-валидации. Из графика можно заключить, что оптимальным значение гиперпараметра c=0.8300000000000001, однако в процессе обучения интерпретатор выдавал предупреждение о том, что данные не являются до конца подготовленными, не отмасштабированными.

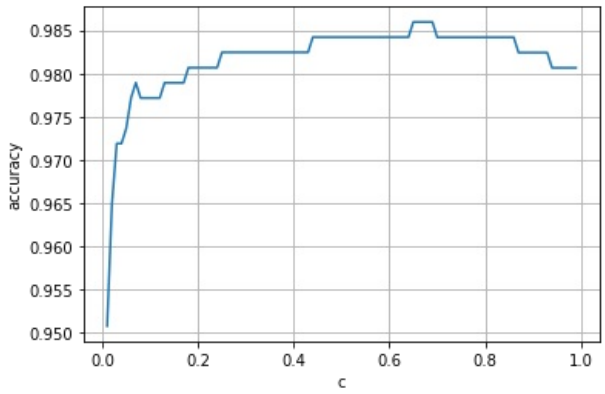
In [60]:
```
scaler = StandardScaler()
X = scaler.fit_transform(X)
```

In [66]:
```
c = {'C': np.arange(0.01,1,0.01)}
model = LogisticRegression(random_state = 241)
gs = GridSearchCV(model, c, scoring='accuracy', cv=kf)
gs.fit(X, y)
```

Out[66]:
```
GridSearchCV(cv=KFold(n_splits=5, random_state=None, shuffle=True),
             estimator=LogisticRegression(random_state=241),
             param_grid={'C': array([0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1 , 0.11,
       0.12, 0.13, 0.14, 0.15, 0.16, 0.17, 0.18, 0.19, 0.2 , 0.21, 0.22,
       0.23, 0.24, 0.25, 0.26, 0.27, 0.28, 0.29, 0.3 , 0.31, 0.32, 0.33,
       0.34, 0.35, 0.36, 0.37, 0.38, 0.39, 0.4 , 0.41, 0.42, 0.43, 0.44,
       0.45, 0.46, 0.47, 0.48, 0.49, 0.5 , 0.51, 0.52, 0.53, 0.54, 0.55,
       0.56, 0.57, 0.58, 0.59, 0.6 , 0.61, 0.62, 0.63, 0.64, 0.65, 0.66,
       0.67, 0.68, 0.69, 0.7 , 0.71, 0.72, 0.73, 0.74, 0.75, 0.76, 0.77,
       0.78, 0.79, 0.8 , 0.81, 0.82, 0.83, 0.84, 0.85, 0.86, 0.87, 0.88,
       0.89, 0.9 , 0.91, 0.92, 0.93, 0.94, 0.95, 0.96, 0.97, 0.98, 0.99])},
             scoring='accuracy')
```

In [68]:
```
plt.plot(c_range, gs.cv_results_['mean_test_score'])
plt.xlabel('c')
plt.ylabel('accuracy')
plt.grid('on')
print("Лучшее значение C: ",gs.best_params_)
print("Лучшее занчение точности: ",gs.best_score_)
```

```
Лучшее значение C:  {'C': 0.65}
Лучшее занчение точности:  0.9859493867411893
```



Вывод из графика

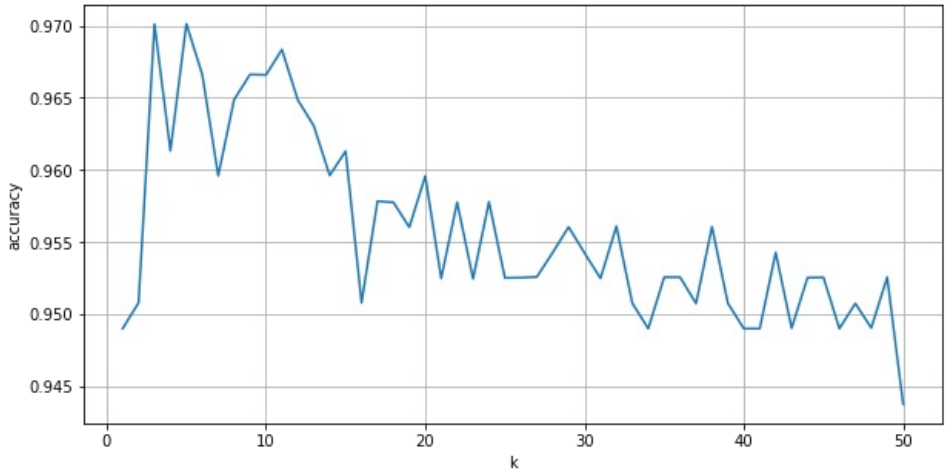
В данном графике отображена точность алгоритма проверенная методом кросс-валидации после проведенного масштабирования выборки. В данном случае интерпритатор уже не

выдает предупрежление, а общая точность повысилась значительно.

Проверка влияния масштабирования на метеод k-ближайших соседей

In [71]:
```
kf = KFold(n_splits=5,shuffle=True)
score = []
accuracy = []
for k in range(1,51):
    model = KNeighborsClassifier(n_neighbors=k)
    score = cross_val_score(model, X,y, cv=kf, scoring = "accuracy")
    accuracy.append(score.mean())
```

In [72]:
```
neighbours = range(1,51)
plt.figure(figsize = [10,5])
plt.plot(neighbours,accuracy)
plt.xlabel('k')
plt.ylabel('accuracy')
plt.grid("on")
```



Вывод из графика

В данном графике отображена точность алгоритма k-блиэайших соседей проверенная методом кросс-валидации после проведенного масштабирования выборки. Общая точность алгоритма также повысилась, а значение k ~ 5

Вывод

В данной работе была проведено пргнозирование больных раком груди по данным медицинских показателей. В работе были использованы линейные модели класссификации, такие как метод k - ближайших соседей и логистическая регрессия. Обе модели показывают достаточно хорошее качество, но только после предварительной их обработки, а именно масштабирования обучающей выборки

In [ ]:
```
```